O. K. TO BE ENTERED, January 12, 2009, /BEN C WANG/

DOCKET NO.: MSFT-2763/305222.1 **PATENT**
Application No.: 10/676,743
Office Action Dated: July 23, 2008

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:
Anandhi Somasekaran, Sergey Chub, Tolga Yildirim, Gueorgui Bonov Chkodrov, Kraig S. Rury, Lucy Ling-Chu Chao, Vladimir Pogrebinsky

Confirmation No.: **7938**

Application No.: **10/676,743** Group Art Unit: **2192**

Filing Date: **September 30, 2003** Examiner: **Ben C. Wang**

For: **Non-Disruptive Business Process Debugging And Analysis**

Mail Stop AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

REPLY PURSUANT TO 37 CFR § 1.116

In response to the Official Action dated **July 23, 2008**, reconsideration is respectfully requested in view of the amendments and/or remarks as indicated below:

- ☐ **Amendments to the Specification** begin on page of this paper.
- ☒ **Amendments to the Claims** are reflected in the listing of the claims which begins on page 2 of this paper.
- ☐ **Amendments to the Drawings** begin on page of this paper and include an attached replacement sheet.
- ☒ **Remarks** begin on page 13 of this paper.

This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Previously Presented) A business process service debugger for remotely debugging a business process service, comprising:

means for establishing a communications connection with a remote computer, wherein the remote computer is implementing the business process service;

means for reading stored state information regarding events related to at least one business process implemented for the business process service;

means for displaying a symbolic representation of the operation of the business process service based on the stored state information; and

means for remotely debugging the business process service using the symbolic representation, communications connection and stored state information.

2. (Previously Presented) The business process service debugger of claim 1, wherein business processes and instances of the business process service other than those being debugged are not disrupted during debugging.

3. (Previously Presented) The business process service debugger of claim 1, wherein the symbolic representation comprises a workflow of at least one business process in the business process service.

4. (Previously Presented) The business process service debugger of claim 1, further comprising means for interacting with the business process service according to a user instruction.

5. (Original) The business process service debugger of claim 1, wherein the stored state information corresponds to a variable assignment within the business process service.

6. (Previously Presented) The business process service debugger of claim 1, wherein the events are historical events that occurred prior to failure of the at least one business process.

7. (Previously Presented) The business process service debugger of claim 1, wherein the stored state information corresponds to message flow data and an order in which run time components performed the one business process as a result of message processing.

8. (Original) The business process service debugger of claim 1, wherein said reading means further comprises means for reading stored business process service configuration information.

9. (Previously Presented) The business process service debugger of claim 1, wherein the events are events that occur prior to an inserted breakpoint in the one business process.

10. (Original) The business process service debugger of claim 1, wherein said debugging means comprises means for detecting a location where the instance is being processed.

11. (Original) The business process service debugger of claim 1, wherein said debugging means comprises means for detecting a location where the instance state is being stored.

12. (Currently Amended) A system for remotely debugging a distributed transactional application, comprising:

a server configured to execute an instance of a business process service comprising a plurality of business processes, thereby generating runtime data;

a client computer configured to execute a debugging user interface (UI) process that establishes a communications connection with the server requests runtime data for at least one of the plurality of business processes, and generates, based on the runtime data, a symbolic representation of the business service process showing any debugging break points specified by a user; and

an interceptor for monitoring the runtime data and, when a specified break point is identified, causing the server to enter or leave a debugging state.

13. (Original) The system of claim 12, further comprising a database for receiving the runtime data and for storing business process service state information.

14. (Previously Presented) The system of claim 13, further comprising a display device for displaying the symbolic representation, and a user input device, wherein the input device is used to specify debugging break points.

15. (Previously Presented) The system of claim 14, wherein the symbolic representation comprises a workflow representative of the program flow of the business process service.

16. (Previously Presented) The system of claim 14, wherein the display device further displays data representative of a message flow of the business process service.

17. (Previously Presented) The system of claim 14, wherein the symbolic representation is presented according to stored state information.

18. (Previously Presented) The system of claim 12, wherein a message box database is coupled between the server and client computer and is configured for communicating debugging requests from the client computer.

19. (Previously Presented) The system of claim 18, wherein the UI process comprises an application program interface for communicating with the message box database.

20. (Previously Presented) The system of claim 18, further comprising a tracking database to receive business process service tracking information, wherein the UI process comprises a UI component for communicating with the tracking database.

21. (Canceled)

22. (Canceled)

23. (Original) The system of claim 12, wherein the interceptor is a component of a computer language that provides stored state tracking information.

24. (Original) The system of claim 12, wherein the UI process detects a location where the instance is being processed.

25. (Original) The system of claim 12, wherein the UI process detects a location where the instance state is being stored.

26. (Previously Presented) A method for debugging an instance of a business process service running on a remote computer, comprising:

generating for display, in a graphical user interface (GUI), a symbolic representation of the business process service based on a correlation of information about the design and execution of the business process service;

receiving a debugging command generated by a user interacting with the GUI;

establishing a direct client connection channel with the remote computer;

causing an interceptor to monitor runtime data generated by the instance of the business process service in accordance with the debugging command;

receiving a runtime request, generated by a user interacting with the GUI, for event information generated by execution of the instance of the business process service;

sending the runtime request to the remote computer for processing by the remote computer.

27. (Original) The method of claim 26, further comprising:
querying a database containing a status of the business process service;
displaying a query result on a display device;
receiving user input with respect to the query result; and
establishing the direct client connection channel in response to the user input.

28. (Original) The method of claim 27, wherein the information contained in the database is instance runtime data.

29. (Original) The method of claim 27, wherein the information contained in the database is instance tracking data.

30. (Previously Presented) The method of claim 26, further comprising:
creating the business process service using a process designer;
saving a business process service configuration and symbolic data in a database as information about the design of the business process service;
displaying the symbolic representation on a display device according to the saved business process service configuration and symbolic data;
generating a runtime request based on the symbolic representation; and
displaying a result of the runtime request on the display device.

31. (Previously Presented) The method of claim 30, wherein the symbolic representation comprises a shape corresponding to an operation in the business process service.

32. (Previously Presented) The method of claim 30, wherein the symbolic representation comprises a workflow representation of the business process service.

33. (Original) The method of claim 30, wherein the saving step takes place in connection with compiling and deploying the business process service.

34. (Original) The method of claim 30, wherein the business process service is implemented in a computer language that provides stored state information.

35. (Previously Presented) The method of claim 26, wherein the debugging command is a break point.

36. (Canceled)

37. (Previously Presented) The method of claim 26, wherein the runtime data is state information.

38. (Original) The method of claim 26, further comprising detecting a location where the instance is being processed.

39. (Original) The method of claim 26, further comprising detecting a location where an instance state is being stored.

40. (Previously Presented) A method in a computer system for displaying on a display device a graphical debugging environment for a business process service, the method comprising:

obtaining design information about the business process service;

obtaining tracking information about execution of the business process service;

generating a symbolic representation of the operation of the business process service according to the design information and tracking information; and

displaying on the display device a graphical debugging environment showing the symbolic representation.

41. (Original) The method of claim 40, further comprising receiving runtime data for the business process service and presenting the runtime data on the display device.

42. (Previously Presented) The method of claim 41, wherein the runtime data comprises historical message flow information including identification of run time messages that were constructed as a result of processing received messages, and further comprises order information pertaining to the order in which different run time components were executed as a result of processing received messages.

43. (Previously Presented) The method of claim 40, wherein the graphical debugging environment enables a user to place a breakpoint in the symbolic representation of the operation of the business process service.

44. (Previously Presented) The method of claim 40, the symbolic representation comprising symbols, wherein the graphical debugging environment also displays information about the symbols.

45. (Previously Presented) The method of claim 40, further comprising receiving input from an input device to place a break point proximate a symbol, and presenting a symbol representing the break point on the symbolic representation.

46. (Previously Presented) A computer-readable storage medium having computer-executable instructions for performing a method for debugging an instance of a business process service running on a remote computer, comprising:

generating for display, in a graphical user interface (GUI), a symbolic representation of the business process service based on a correlation of information about the design and execution of the business process service;

receiving a debugging command generated by a user interacting with the GUI;

establishing a direct client connection channel with the remote computer;

causing an interceptor to monitor runtime data generated by the instance of the business process service in accordance with the debugging command;

receiving a runtime request; and

sending the runtime request to the remote computer for processing by the remote computer.

47. (Previously Presented) The computer-readable storage medium of claim 46, wherein the method further comprises:

querying a database containing a status of the business process service;

displaying a query result on a display device;

receiving user input with respect to the query result; and

establishing the direct client connection channel in response to the user input.

48. (Previously Presented) The computer-readable storage medium of claim 47, wherein the information contained in the database is instance runtime data.

49. (Previously Presented) The computer-readable storage medium of claim 47, wherein the information contained in the database is instance tracking data.

50. (Previously Presented) The computer-readable storage medium of claim 46, wherein the method further comprises:

creating the business process service using a process designer;

saving business process service configuration data in a database as information about the design of the business process service;

displaying the symbolic representation on a display device according to the saved business process service configuration data;

generating a runtime request based on the symbolic representation; and

displaying a result of the runtime request on the display device.

51. (Previously Presented) The computer-readable storage medium of claim 50, wherein the symbolic representation comprises a shape corresponding to an operation in the business process service.

52. (Previously Presented) The computer-readable storage medium of claim 50, wherein the symbolic comprises a workflow representation of the business process service.

53. (Previously Presented) The computer-readable storage medium of claim 50, wherein the saving step takes place in connection with compiling and deploying the business process service.

54. (Previously Presented) The computer-readable storage medium of claim 50, wherein the business process service is implemented in a computer language that provides stored state information.

55. (Previously Presented) The computer-readable storage medium of claim 50, wherein the debugging command is a break point.

56. (Previously Presented) The computer-readable storage medium of claim 50, wherein the debugging command is a request for data regarding an instance of the business process service.

57. (Previously Presented) The computer-readable storage medium of claim 56, wherein the runtime data is state information.

58. (Previously Presented) The computer-readable storage medium of claim 46, wherein the method further comprises detecting a location where the instance is being processed.

59. (Previously Presented) The computer-readable storage medium of claim 46, wherein the method further comprises detecting a location where an instance state is being stored.

60. (Previously Presented) A computer-readable storage medium having computer-executable instructions for performing a method for displaying on a display device a graphical debugging environment for a business process service, the method comprising:

obtaining design information about the business process service;

obtaining configuration information about the business process service;

generating a symbolic representation of the operation of the business process service according to the design information and configuration information; and

displaying on the display device a graphical debugging environment showing the symbolic representation.

61. (Previously Presented) The computer-readable storage medium of claim 60, wherein the method further comprises receiving runtime data for the business process service and presenting the runtime data on the display device.

62. (Previously Presented) The computer-readable storage medium of claim 61, wherein the runtime data comprises message flow information.

63. (Previously Presented) The computer-readable storage medium of claim 60, wherein the graphical debugging environment enables a user to place a breakpoint in the symbolic representation of the operation of the business process service.

64. (Previously Presented) The computer-readable storage medium of claim 60, the symbolic representation comprising symbols, wherein the graphical debugging environment also displays information about the symbols.

65. (Previously Presented) The computer-readable storage medium of claim 60, wherein the method further comprises receiving input from an input device to place a break point proximate a symbol, and presenting a symbol representing the break point on the symbolic representation.

REMARKS

In summary, 62 claims numbered 1-20, 23-35, and 37-65 are pending. Claims 1, 12, 26, 40, 46, and 60 are independent. Claim 12 is hereby amended without adding new matter. Claims 1-20 and 23-25 are rejected under 35 U.S.C. § 103. Claims 40, 41, 43-45, 60, 61, and 63-65 are rejected under 35 U.S.C. § 102. Claims 26-35, 37-39, 42, 46-59, and 62 are rejected under 35 U.S.C. § 103. Applicants respectfully traverse all rejections. Reconsideration in view of the foregoing amendments and following remarks is respectfully requested.

Telephone Conversation With Examiner

Examiner Wang is thanked for the telephone conversations conducted on October 9, 2008 and October 22, 2008. During the conversation conducted on October 9, 2008, cited art was discussed. The teaching away against combining the cited art was discussed. Differences between the cited art and the claims were discussed. Examiner Wang stated that he would consider Applicants' remarks and call Applicants' representative to discuss further. After the conversation conducted on October 9, 2008, Examiner Wang informed Applicants' representative that he found additional art and suggested another telephone conversation to discuss same. During the second conversation conducted on October 22, 2008, it was agreed that Applicants' representative would submit the instant response and receive a non-final office action based on the new art found by Examiner Wang. Again, Applicants' representative thanks Examiner Wang for his cooperation in this matter.

Previous Remarks

The Office Action withdrew previous rejections, but asserts new grounds of rejection. Even so, the Examiner made the Office Action final. While the Office Action alleges that Applicants' previous remarks are moot in view of the new grounds of rejection, Applicants point out that their previous remarks are not moot because the Office Action rejects 35 of the 62 claims based in part on combinations involving the Roxburgh reference. Applicants' previous remarks pointed out that

Roxburgh teaches away from the claimed subject matter, which means that it cannot be combined with other references to try to overcome the teaching away. Specifically, Applicants said:

Roxburgh teaches away from the claimed subject matter . . . Roxburgh, § "Debugging Components in Schedules" at p. 16 ("schedules themselves can't be loaded into a visual environment and debugged."); Roxburgh, § "Debugging Schedules" at p. 14 ("Microsoft BizTalk Server does not provide a graphical debugging facility for orchestration schedules. However, remember that orchestration schedules represent a different kind of executable process from traditional short-lived synchronous processes, so the traditional debugging model alone is not effective. . . . When the orchestration schedule is compiled and then executed, typically multiple instances of the schedule will be initiated as individual (long-running) executable processes.").

"A reference may be said to teach away when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the applicant." In re Gurley, 27 F.3d 551, 553 (Fed. Cir., 1994).

Clearly, since Microsoft personnel such as Roxburgh thought it was not possible, a person of ordinary skill would be discouraged from the claimed subject matter. This disclosure in Roxburgh teaching away from the claimed invention cannot be ignored because each "prior art reference must be considered in its entirety, i.e., as a whole, including portions that would lead away from the claimed invention." M.P.E.P. § 2141.02 VI. "It is improper to combine references where the references teach away from their combination." M.P.E.P. § 2145 X.D.2.

Therefore, Applicants respectfully submit that their prior remarks are not moot in view of the new grounds of rejection and should have been considered by the Examiner.

Rejection of Claims 1-20 and 23-25 under 35 U.S.C. § 103(a)

Claims 1-20 and 23-25, including independent claims 1 and 12, are rejected under 35 U.S.C. § 103(a) as being unpatentable over (1) "BizTalk Orchestration: Transactions, Exceptions, and

Debugging," Feb. 2001, authored by Roxburgh (hereinafter referred to as "Roxburgh") in combination with either or both: (2) U.S. Patent No. 7,051,316, issued to Charisius et al. (hereinafter referred to as "Charisius"); and (3) "BizTalk Unleashed," 1st ed., Feb. 2002, authored by Adams *et al.* (hereinafter referred to as "Adams"). (Office Action, pp. 3-18). Applicants respectfully traverse the rejections.

The citations made to the references in the Office Action either do not disclose what the Office Action says they do or they actually teach away from the claimed subject matter.

First, in Charisius, the Office Action points to passages with keywords. However, the substance of those passages does not teach or suggest the claimed subject matter, e.g., "displaying a <u>symbolic representation</u> of the operation of the business process service <u>based on</u> the <u>stored state information [e.g. runtime data]</u>; and . . . remotely <u>debugging</u> the business process service <u>using</u> the <u>symbolic representation</u> . . . <u>and stored state information</u>." Claim 1.

In response to this subject matter in claim 1, the Office Action cites column 28, line 48 to column 29, line 3 for the word "symbol" then skips to Figure 24 and column 26, lines 59-64 for the words "state" and "events" then skips to Figure 61 for the word "debug" and then column 38, lines 8-10 for the word "remote." Keywords or not, the substance of these citations does not teach or suggest the substance of claim 1.

Looking at Figure 24 and the cited paragraph in column 28, it is very clear that it is not "a symbolic representation of the operation of the business process service based on the stored state information [e.g. runtime data]." To begin with, there are no symbols representing operations/steps carried out by the process, i.e., "a symbolic representation of the operation of the . . . process." The so-called "graphical representation of the source code" is a textual list. The so-called "symbols" only indicate the method types that the textual list falls under. There is no "symbolic representation of the operation of [a] business process." The meaning of the terms used in Charisius is not the same as claimed.

Furthermore, the so-called graphical representation and symbols, even if they were a "symbolic representation of the operation of [a] business process," are not based on stored state information. Charisius does not even contemplate that. Nonetheless, the Office Action tries to make up for that lack of disclosure by citing to a mere mention that the textual list can include a state management callback method used to tell an EntityBean "when certain events are to occur." This statement about what a listed method can do does not magically translate Figure 24 into a symbolic representation of a process based on stored state information (e.g. actual runtime data). Charisius' displayed information does not even include stored state information, let alone show a symbolic representation constructed from such information. The citations are empty, cited only for their mention of keywords. A proper rejection must be based on actual disclosure of substance, not a mere cobbling together of keywords such as "state" here, "symbol" there.

Since there is no symbolic representation of a process based on stored state information, there is also no remote debugging that relies on a symbolic representation of process operation and stored state information. Figures 60 and 61 show the debugging user interface and it does not include a symbolic representation of process operation, let alone one based on stored state information (e.g. runtime data). It is observed that the Office Action does not offer any citation as allegedly disclosing "remotely debugging the business process service using the symbolic representation . . . and stored state information." An alleged partial disclosure by Charisius is obviously not enough to even make a prima facie rejection. And this is just the end of the first point proving the rejection to be without merit.

Second, as pointed out in previous remarks, Roxburgh teaches away from the claimed subject matter of claims 1-7 and 9-11. See, e.g., Roxburgh, § "Debugging Components in Schedules" at p. 16 ("schedules themselves can't be loaded into a visual environment and debugged."); Roxburgh, § "Debugging Schedules" at p. 14 ("Microsoft BizTalk Server does not provide a graphical debugging facility for orchestration schedules. However, remember that orchestration schedules represent a different kind of executable process from traditional short-lived synchronous processes, so the traditional debugging model alone is not effective. . . .

When the orchestration schedule is compiled and then executed, typically multiple instances of the schedule will be initiated as individual (long-running) executable processes.").

"A reference may be said to teach away when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the applicant." In re Gurley, 27 F.3d 551, 553 (Fed. Cir., 1994).

Clearly, since Microsoft personnel such as Roxburgh thought it was not possible, a person of ordinary skill would be discouraged from the claimed subject matter. This disclosure in Roxburgh teaching away from the claimed subject matter cannot be ignored because each "prior art reference must be considered in its entirety, i.e., as a whole, including portions that would lead away from the claimed invention." M.P.E.P. § 2141.02 VI. "It is improper to combine references where the references teach away from their combination." M.P.E.P. § 2145 X.D.2.

It is improper for the Office Action to ignore the part of Roxburgh that teaches away from the claimed subject matter in favor of part of a different reference. To ignore this teaching away in Roxburgh is to fail to read Roxburgh as a whole and to use improper hindsight of the claimed subject matter.

For at least the foregoing two reasons, the rejection of claim 1 is without merit. The foregoing remarks apply equally well to each of claims 1-20 and 23-25. Accordingly, Applicants respectfully request withdrawal of the rejection.

Rejection of Claims 40, 41, 43-45, 60, 61 and 63-65 under 35 U.S.C. § 102(b)

Claims 40, 41, 43-45, 60, 61 and 63-65, including independent claims 40 and 60, are rejected under 35 U.S.C. § 102(e) as being anticipated by Charisius. (Office Action, pp. 18-23). Applicants respectfully traverse the rejection.

The foregoing remarks apply equally well here. The Office Action makes the same citation against claim 40 that is made against claim 1. Claims 40 and 60 say, "generating a symbolic representation of the operation of the business process service according to the design information ["about the business process service"] and tracking [or "configuration"] information ["about execution of the business process service"]; and displaying . . . a graphical debugging environment showing the symbolic representation."

The Office Action cites the same portions of Charisius cited against claim 1, i.e., column 28, line 46 to column 29, line 3; Figure 24 and column 26, lines 59-64 of Charisius. As previously pointed out, these citations fall far short. Nowhere does Charisius disclose a symbolic representation of the operation of a process, let alone a symbolic representation constructed from both design and execution information. The citations are obviously based on keywords alone and do not remotely convey the substance of the claimed subject matter. Therefore, the citations and the arguments they are made for are without merit.

Independent claims 40 and 60, as well as the claims that depend on them are clearly patentable over Charisius. Accordingly, Applicants respectfully request withdrawal of the rejection.

Rejection of Claims 26-35, 37-39, 42, 46-59, and 62 under 35 U.S.C. § 103(a)

Claims 26-35, 37-39, 42, 46-59 and 62, including independent claims 26 and 46, are rejected under 35 U.S.C. § 103(a) as being unpatentable over Charisius in view of Adams and, in the case of claims 27-35 and 37-39, Roxburgh. Applicants respectfully traverse the rejection.

The foregoing remarks apply equally well here. The Office Action makes the same citation against claims 26 and 46 that is made against claims 1, 12, 40 and 60. Claims 26 and 46 say, "generating . . . a symbolic representation of the business process service based on a correlation of information about the design and execution of the business process service"

The Office Action cites the same portions of Charisius cited against claims 1, 12, 40 and 60, i.e., column 28, line 46 to column 29, line 3; Figure 24 and column 26, lines 59-64 of Charisius. As previously pointed out, these citations fall far short. Nowhere does Charisius disclose a symbolic representation of the operation of a process, let alone a symbolic representation constructed from both design and execution information. The citations are obviously based on keywords alone and do not remotely convey the substance of the claimed subject matter. Therefore, the citations and the arguments they are made for are without merit.

Independent claims 26 and 46, as well as the claims that depend on them, are clearly patentable over Charisius combined with Adams and Roxburgh, which do not make up for the lack of disclosure by Charisius. As such, Applicants respectfully request withdrawal of the rejection.

Amendments made herein as well as amendments previously made are without abandonment of subject matter. Applicant expressly reserves the right to, in the pending application or any application related thereto, reintroduce any subject matter removed from the scope of claims by any amendment and introduce any subject matter not present in current or previous claims.

CONCLUSION

In view of the foregoing remarks and amendments, it is respectfully submitted that this application is in condition for allowance. Reconsideration of this application and an early Notice of Allowance are requested. Applicants desire to hold a telephone interview with the Examiner and his supervisor following their review of this reply.

Date: October 22, 2008

/Joseph F. Oriti/
Joseph F. Oriti
Registration No. 47,835

Woodcock Washburn LLP
Cira Centre
2929 Arch Street, 12th Floor
Philadelphia, PA 19104-2891
Telephone: (215) 568-3100
Facsimile: (215) 568-3439